# FastFed for Governance ()

## Overview

FastFed, by the final draft, will define a “provider” somewhat agnostically in order to support various provider types. Currently, provider types for FastFed example stories (Alice) and demo implementations of the FastFed standard are:

* Identity Provider (IdP) - ADFS, Okta, Google, Ping, etc.
* Service Provider (SP) – Box, Salesforce, etc.

FastFed, however, will not be limited to these. SailPoint proposes another provider type:

* Governance Provider (GP) - SailPoint, Okta, future CAEP Providers, etc.

This document aims to outline what SailPoint feels are the differences for a GP verses other provider types, what a FastFed for Governance flow might entail given those differences, and pose questions for the working group.

## Current Standard

The current FastFed standard clearly defines the FastFed process for automating the process of configuring SSO for a SP and an IdP. Some aspects of this will be discussed, but the assumption is that the reader is familiar enough with the working standard.

The current standard requires a “start” endpoint on the IdP side in which a service provider (relying party) would make a GET request to initiate the FastFed process. This endpoint, in most cases, would be initiated by some UX in the application if the SP supports FastFed.

Although not mandatory, it makes most sense to initiate the process from the SP because there is callback information that must be given to the IdP for the next step in the handshake and the SP would inherently have this information. Each entity in the setup (SP and IdP) needs to know specific details about each other to complete the FastFed process and create the SSO relationship. A SP that “wants” to use SSO for authentication needs a mutual relationship and understanding with an IdP and as such, the IdP requires information specific to the SP. With governance, however, this is not the case.

## Governance

#### Overview

In the GP scenario, a SP (application) doesn’t specifically “want” to be governed. It may ALLOW itself to be governed by means of an already existing infrastructure for obtaining application API credentials and a SCIM endpoint or other API (REST, etc.) for provisioning, but there is no additional information required by the SP and as far as the application is concerned, the GP is like any other API client.

FastFed Governance would help to simplify and automate the process of configuring the SP to be governed, but the process would differ from the existing flow used during SSO configuration automation. Because of this, questions arise as to whether modifications or deviations from the current standard would be necessary for GPs.

#### SCIM Provisioning

The FastFed standard uses well-known SAML standards for information about each entity. FastFed for Governance would also require such a standard in order to generically provide the ability to provision, deprovision, and govern a SP. As a result of this requirement, the SP would be required to implement a SCIM endpoint supporting, at a minimum, a “User” profile.

#### General Assumptions

1. There is no requirement that a SP support SSO in order to use the Fed for Governance process.
2. If a SP does not support SSO and another authentication mechanism is used, it is assumed that the SP will properly redirect to the requested “start” endpoint after authentication, with the original URL and associated parameters intact.
3. There would be a different “start” endpoint for each provider type, where applicable. For the SSO scenario, this would be the IdP. For the Governance scenario, this would be the SP.
4. It is assumed that OAuth will be the only standard supported for authentication in order to potentially support Dynamic OAuth client registration.

#### How it Works

Since the SP has no concept of who or what is governing it, or even that it is being governed, it makes sense to initiate the governance process from the GP side. It would be up to the SP to implement a FastFed metadata endpoint. The GP would discover this endpoint through its WebFinger metadata for a given account(email) and the FastFed “rel” identifier. The GP is responsible for knowing what SP’s it can govern and how it can govern them (SCIM, API, etc.) based on FastFed metadata defined profiles or other company specific criteria.

If a SP provides a FastFed governance “start” endpoint, a UX could be updated accordingly for the GP during setup of the SP’s governance configuration.

###### Discovery Mechanism – (WebFinger)

Having the GP initiate the process also makes it easier multiple instances of the same SP to be governed properly. An example of this might be having two AWS accounts being managed by a GP. These would be represented as separate applications in the GP. As such, in addition to requiring an email address for a WebFinger lookup, a tenant or instance information would also be required. Using the “rel” [http://openid.net/specs/fastfed/1.0/provider](http://openid.net/specs/fastfed/1.0/provider/), the GP can discover all of the SP’s available to Alice.

See more information regarding discovery in the flow steps below.

# Flow

Given the information in the previous sections, using our good friend Alice, the flow/story would be something like:

Step 1

Alice logs in to her GP ([www.governance.com](http://www.governance.com)). She is very excited to set up governance for a SP ([www.serviceprovider.com](http://www.serviceprovider.com)). The smile on her face says it all. She visits the section of the GP where she can create a SP to be governed and she is overwhelmed with joy when she sees the “FastFed Governance” button because she knows FastFed just made her life that much easier. Alice clicks the button. She is asked for her email address. She enters [alice@company.com](mailto:alice@company.com) She clicks “OK”.

Step 2

To be consistent with the FastFed standard, the GP uses the entered information to discover the “start” endpoint for the email and SP instance entered. Webfinger returns something like:

|  |
| --- |
| GET /  ?resource=acct%3Aalice%40company.com  &rel=https://www.serviceprovider.com  HTTP/1.1  Host: webfinger.\_well\_known.company.com |
| HTTP/1.1 200 OK  Content-Type: application/jrd+json    {  "subject": "acct:alice@company.com",  "links": [  {  "rel": " http://openid.net/specs/fastfed/1.0/provider ",  "href": <https://prod-aws.serviceprovider.com/fastfed>,  "titles" :  {  "en-us" : "Production AWS"  }  },  {  "rel": " http://openid.net/specs/fastfed/1.0/provider ",  "href": [https://dev-aws.serviceprovider.com/fastfed](https://dev-aws.serviceprovider.com/fastfed/iga),  "titles" :  {  "en-us" : "Development AWS"  }  }  ]  } |

Step 3

Using the WebFinger response, the GP shows all matching links based on the titles for her language and asks Alice which SP she will be setting up for governance. In this case, she would see, in a dropdown,

Production AWS

Development AWS

She chooses “Production AWS”. The GP now makes a GET request to the URL (<https://prod-aws.serviceprovider.com/fastfed>)stored in the “href” value for the matching record with that title. This returns the SP’s governance metadata.

|  |
| --- |
| GET /fastfed/ HTTP/1.1  Accept: application/json  Host: www.governance.com |
| HTTP/1.1 200 OK  Content-Type: application/jrd+json    {  "service\_provider": {  "provider\_domain\_name": "serviceprovider.com",  "tenant\_id": "prod-aws",  "jwks\_uri": "https://serviceprovider.com/fastfed/keys",  "display\_name": "Production AWS Service Provider",  "display\_images": {  "logo\_uri": "https://serviceprovider.com/images/logo.png",  "icon\_uri": "https://serviceprovider.com/images/icon.png"  }  "capabilities": {  "authentication\_profiles": [  "urn:ietf:params:fastfed:1.0:authentication:OAuth2:ClientCredentials",  ],  "provisioning\_profiles":[  "urn:ietf:params:fastfed:1.0:provisioning:SCIM:FullLifeCycle"  ],  "schemas": [  "urn:ietf:params:scim:schemas:core:2.0:User"  ],  "handshake\_signing\_alg": [  "RS256"  ]  }  "fastfed\_handshake\_start\_url":  "https://prod-aws.serviceprovider.com/fastfed/iga/start",  }  } |

If WebFinger does not return any information or discovery is not enabled (or kept current), Alice is still able to enter the FastFed metadata endpoint url manually.

Step 4

The GP looks up the “fastfed\_handshake\_start\_url” in the FastFed metadata and opens a new tab for Alice with that URL and additional information on the querystring. This is exactly how FastFed handles this step during the SSO/Idp setup. If SSO is being used, Alice logs in using her IdP. Otherwise, she logs in to the SP using its existing login mechanism.

|  |
| --- |
| HTTP/1.1 302 Found  Location: https://prod-aws.serviceprovider.com/fastfed/iga/start?  expiration= 1547414843  &metadata\_uri=  https%3A%2F%2Fwww.governance.com%2Ffastfed%2Fprovider-metadata |

Step 5

Alice is still excited but pauses to take a restroom break before moving on. Six minutes and 34 seconds later, Alice returns to her desk. It is possible she also stopped for a glass of water. Now that she is back, unbeknownst to her, the FastFed process has continued. Using the information specified on the querystring, the SP has called to the GP using the “metadata\_uri” and retrieves the GP’s metadata.

|  |
| --- |
| GET /fastfed/provider-metadata HTTP/1.1  Accept: application/json  Host: www.governance.com |
| HTTP/1.1 200 OK  Content-Type: application/json  {  “identity\_provider”: {  …  }  "governance\_provider": {  "provider\_domain\_name": "governance.com",  "tenant\_id": "www",  "display\_name": "Governance Provider",  "display\_images": {  "logo\_uri": "https://governance.com/images/logo.png",  "icon\_uri": "https://governance.com/images/icon.png"  }  "capabilities": {  "authentication\_profiles": [  "urn:ietf:params:fastfed:1.0:authentication:OAuth2:ClientCredentials",  ],  "handshake\_signing\_alg": [  "RS256"  ]  }  "fastfed\_handshake\_finish\_uri":  "https://www.governance.com/fastfed/finish",  } |

Note that the FastFed metadata has been combined. The “identity\_provider” is optional as well as “governance provider” but at least one must be specified.

Step 6

Using whatever information the SP needs from the GP’s FastFed metadata, it creates a clientId/secret (refresh token or other mechanism for authentication) to give the GP the ability to use the SCIM endpoint. The service provider then POST the following information to the GP at the URL specified in the “fastfed\_handshake\_finish\_uri” value.

|  |
| --- |
| {  "iss": "serviceprovider.com",  "sub": "prod-aws",  "aud": " serviceprovider.com",  "schemas": [  "urn:ietf:params:scim:schemas:core:2.0:User"  ],  "authentication\_profiles": [  "urn:ietf:params:fastfed:1.0:authentication:OAuth2:ClientCredentials  "  ],  "provisioning\_profiles": [  "urn:ietf:params:fastfed:1.0:provisioning:SCIM:FullLifeCycle"  ],  "urn:ietf:params:fastfed:1.0:authentication:SAML:Basic": {  "scim\_base\_uri":  "https://prod-aws.serviceprovider.com/scim",  },  "oauth\_token": {  "access\_token": "MTQ0NjJkZmQ5OTM2NDE1ZTZjNGZmZjI3",  "token\_type": "bearer",  "refresh\_token": "IwOGYzYTlmM2YxOTQ5MGE3YmNmMDFkNTVk",  "expires\_in": 3600  }  } |

As with FastFed SSO, this will be encoded and signed as a JWT.

Step 7

The GP receives the SP’s JWT and verifies it. Using this information, the GP has enough information to create the governance record for the SP. Using the OAuth information (or whatever credentials are supported), and the schemas the SP supports, the GP can now provision using the SCIM endpoint.

Since the SP doesn’t have any concern about whether the GP is now able to govern it successfully, at this point, the process is completed and Alice sees a message indicating so. She takes a deep breath knowing she saved a significant amount of time and plans on using the time she saved to take up a new hobby, possibly Cricket, but since our story ends here, we may never know.

|  |  |  |
| --- | --- | --- |
| Brian Rose | V1.0 | 09/12/2019 |
| Brian Rose | V1.1 | 09/16/2019 |